Switched Probabilistic I/O Automata

Ling Cheung[1], Nancy Lynch[2], Roberto Segala[3], and Frits Vaandrager[1]

[1] Nijmegen Institute for Computing and Information Sciences
University of Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
{lcheung,fvaan}@cs.kun.nl*
[2] MIT Computer Science and Artificial Intelligence Laboratory
Cambridge, MA 02139, USA
lynch@theory.csail.mit.edu**
[3] Dipartimento di Informatica, Università di Verona
Strada Le Grazie 15, 37134 Verona , Italy
roberto.segala@univr.it***

Abstract. A *switched probabilistic I/O automaton* is a special kind of probabilistic I/O automaton (PIOA), enriched with an explicit mechanism to exchange control with its environment. Every closed system of switched automata satisfies the key property that, in any reachable state, at most one component automaton is active. We define a trace-based semantics for switched PIOAs and prove it is compositional. We also propose *switch extensions* of an arbitrary PIOA and use these extensions to define a new trace-based semantics for PIOAs.

1 Introduction

Probabilistic automata [Seg95,Sto02] constitute a mathematical framework for modeling and analyzing probabilistic systems, specifically, systems consisting of asynchronously interacting components capable of nondeterministic and probabilistic choices. This framework has been successfully adopted in the studies of distributed algorithms [LSS94,PSL00,Agg94] and practical communication protocols [SV99].

An important part of such a framework is a notion of *visible behavior* of system components. This is used to derive implementation and equivalence relations among components. For example, one can define the visible behavior of a nondeterministic automaton to be its set of *traces* (i.e., sequences of visible actions that arise during executions of the automaton [LT89]). This induces

* Supported by DFG/NWO bilateral cooperation project 600.050.011.01 Validation of Stochastic Systems (VOSS).
** Supported by DARPA/AFOSR MURI Award #F49620-02-1-0325, MURI AFOSR Award #F49620-00-1-0327, NSF Award #CCR-0326277, and USAF, AFRL Award #FA9550-04-1-0121.
*** Supported by MURST project Constraint-based Verification of reactive systems (CoVer).

an implementation (resp. equivalence) relation on nondeterministic automata, namely inclusion (resp. equality) of sets of traces.

Perhaps the most important property of an implementation relation is *compositionality*: if P implements Q, then for every context R, one should be able to infer that $P \| R$ implements $Q \| R$. This property facilitates correctness proofs of complex systems by reducing properties of a large system to properties of smaller subsystems. In the setting of security analysis, for instance, compositionality ensures that plugging secure components into a security preserving context results again in a secure component [Can01].

Generalizing the notion of traces, Segala [Seg95] defines the visible behavior of a probabilistic automaton as its set of *trace distributions*, where each trace distribution is induced by a probabilistic *scheduler* which resolves all nondeterministic choices. This gives rise to implementation and equivalence relations as inclusion and equality of sets of trace distributions, respectively. It turns out that this notion of implementation relation is not compositional. A simple counterexample is illustrated in Figure 1.

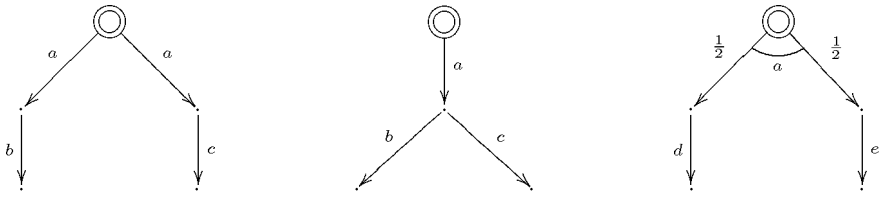

Fig. 1. Probabilistic automata Early, Late and Toss

As their names suggest, automaton Early forces its scheduler to choose between b and c as it chooses one of the two available a-transitions, whereas automaton Late allows its schedulers to make this decision after the a-transition. Clearly, these two automata have the same set of trace distributions, but they can be distinguished by the context Toss. The composed system Late $\|$ Toss has a trace distribution D_0 that assigns probability $\frac{1}{2}$ to each of these traces: adb and aec . Such total correlations between actions d and b, and between actions e and c, cannot be achieved by the composite Early $\|$ Toss.

Inspired by this example, we establish in [LSV03] that the coarsest precongruence refining trace distribution preorder coincides with the probabilistic simulation preorder. In other words, probabilistic contexts are capable of exposing internal branching structures of other components.

Aside from its inspirational merits, this example reveals an unsatisfactory aspect of the composition mechanism of probabilistic automata. Namely, nondeterministic choices are resolved after the two automata are composed, allowing the global scheduler to make decisions in one component using state information of the other. This phenomenon can be viewed as a form of "information leak-

age": the global scheduler channels private information from one component to the other, in particular, from Toss to Late.

In this paper, we present a composition mechanism where local scheduling decisions are based on strictly local information. That is, (i) local nondeterministic choices of each component are resolved by that component alone; (ii) global nondeterministic choices (i.e., inter-component choices) are resolved by some independent means. To address the first issue, we introduce an input/output distinction to our model and pair each automaton with an *input/output scheduler*. For the second, we introduce a control-passage[1] mechanism, which eliminates global scheduling conflicts.

Before describing our model in greater detail, we take a quick look at related proposals in the existing literature. (We refer to [SV04] for a comparative study of various probabilistic models.) For purely synchronous, variable-based models, global nondeterministic choices are resolved by "avoidance": in each transition of the global system, all components may take a step. This intrinsic feature of synchronous models allows De Alfaro, Henzinger and Jhala [dAHJ01] to successfully define a compositional, trace-based semantics for their model of probabilistic reactive modules. For asynchronous models such as probabilistic automata, global nondeterministic choices must be resolved explicitly in order to assign a probability mass to each possible interleaving of actions. Wu, Smolka and Stark [WSS94] propose a compositional model based on probabilistic input/output automata. In that model, global nondeterminism is resolved by a "race" among components: each component draws a delay from an exponential distribution (thus leaving the realm of discrete distributions). Assuming independence of these random draws, the probability of two components drawing the same delay is zero, therefore there is almost always a unique winner.

In this paper, we introduce the model of *switched probabilistic I/O automata* (or *switched automata* for short). This augments the probabilistic I/O automata model with some additional structures and axioms. In particular, we add a predicate active on the set of states, indicating whether the automaton is active or inactive. We require that locally controlled actions are enabled only if the automaton is active. In other words, an inactive automaton must be quiescent and can only accept inputs from the environment.

A switched automaton changes its activity status by performing special control input and control output actions. Control inputs switch the machine from inactive to active and vice versa for control outputs. All other actions must leave the activity status unchanged. It is important that all control communications are "handshakes": at most two components may participate in a synchronization labeled by a control action. Together with an appropriate initialization condition, this ensures that at most one component is active at any point of an execution. Intuitively, we model a network of processes passing a single token among them,

[1] Throughout this paper, the term *control* is used in the spirit of "control flow" in sequential programming: a component is said to possess the control of a system if it is scheduled to actively perform the next action. This should not be confused with the notion of controllers for plants, as in control theory.

with the property that a process enables a locally controlled transition if and only if it possesses the token.

The main technical result of this paper is compositionality of a trace-based semantics for switched probabilistic I/O automata (Section 6, Theorem 1). Sections 2 and 3 are devoted to the basic theory. There we introduce new technical notions such as I/O schedulers, scheduled automata and parallel composition of scheduled automata. In Section 4, we define pseudo probabilistic executions and pseudo trace distributions for automata with open inputs, and prove important projection and pasting results. Section 5 treats two standard operators: renaming and hiding. In Section 7, we propose the notion of *switch extensions* for PIOAs, which can be used to derive a new form of composition for the original PIOA model. Concluding discussions follow in Section 8. Due to space constraints, we have omitted many proofs. These can be found in a full version of this paper available at
`http://www.cs.kun.nl/ita/publications/papers/fvaan/switched.html`.

2 Preliminaries

In this section, we define probabilistic I/O automata and some related notions. This is a straightforward combination of the Input/Output Automata model of Lynch and Tuttle [LT89] and the Simple Probabilistic Automata model of Segala [Seg95].

A *discrete probability (resp. sub-probability) measure* over a set X is a measure μ on $(X, 2^X)$ such that $\mu(X) = 1$ (resp. $\mu(X) \leq 1$). With slight abuse of notation, we write $\mu(x)$ for $\mu(\{x\})$. The set of all discrete probability measures over X is denoted $\mathsf{Disc}(X)$; similarly for $\mathsf{SubDisc}(X)$. Moreover, we use $\mathsf{Supp}(\mu)$ to denote the *support* of a discrete measure μ: the set of elements in X to which μ assigns nonzero measure. Given $x \in X$, the *Dirac distribution* on x is the unique measure assigning probability 1 to x, denoted $(x \mapsto 1)$.

A *probabilistic I/O automaton (PIOA)* P consists of:

- a set $\mathsf{States}(P)$ of states and a *start state* $s^0 \in \mathsf{States}(P)$;
- a set $Act(P)$ of action symbols, partitioned into: I (*input* actions), O (*output* actions) and H (*hidden* actions);
- a transition relation $\rightarrow \subseteq \mathsf{States}(P) \times Act(P) \times \mathsf{Disc}(\mathsf{States}(P))$.

An action is *visible* if it is not hidden. It is *locally controlled* if it is non-input (i.e., either output or hidden); we define $L := O \cup H$. We write $s \xrightarrow{a} \mu$ for $\langle s, a, \mu \rangle \in \rightarrow$, and $s \xrightarrow{a} s'$ if there exists μ with $s \xrightarrow{a} \mu$ and $s' \in \mathsf{Supp}(\mu)$. A state is *quiescent* if it enables only input actions. A PIOA is *closed* if its set of input actions is empty. As with I/O automata, we always assume *input enabling*: $\forall s \in \mathsf{States}(P)\ \forall a \in I\ \exists \mu : s \xrightarrow{a} \mu$.

An *execution* of P is a (possibly finite) sequence $p = s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_2 \ldots$, such that:

- each s_i (resp., a_i, μ_i) denotes a state (resp., action, distribution over states);

- $s_0 = s^0$ and, if p is finite, then p ends with a state;
- for each non-final i, $s_i \xrightarrow{a_{i+1}} \mu_{i+1}$ and $s_{i+1} \in \mathsf{Supp}(\mu_{i+1})$.

In some literature, executions are defined to be sequences of states and actions in alternating fashion, thus omitting the target distributions. We adopt the current style for a more straightforward generalization to probabilistic executions.

Given a finite execution p, we use $\mathsf{last}(p)$ to denote the last state of p. A state s is *reachable* if there exists a finite execution p such that $\mathsf{last}(p) = s$. We write $\mathsf{Exec}(P)$ for the set of all executions of P and $\mathsf{Exec}^{<\omega}(P)$ for the set of finite executions. Given an execution p, the sequence of visible action symbols in p is called the *(visible) trace* of p, denoted $\mathsf{tr}(p)$.

A finite set of PIOAs $\{P_1, \ldots, P_n\}$ is said to be *compatible* if for all $i \neq j$, $O_i \cap O_j = Act(P_i) \cap H_j = \emptyset$. Such a set is *closed* if $\bigcup_{1 \leq i \leq n} I_i \subseteq \bigcup_{1 \leq i \leq n} O_i$. We define $P = \|_{1 \leq i \leq n} P_i$ as usual with synchronization of shared actions:

- $\mathsf{States}(P) := \prod_{1 \leq i \leq n} \mathsf{States}(P_i)$ and the start state of P is $\langle s_1^0, \ldots, s_n^0 \rangle$;
- $I := \bigcup_{1 \leq i \leq n} I_i \setminus \bigcup_{1 \leq i \leq n} O_i$, $O := \bigcup_{1 \leq i \leq n} O_i$, and $H := \bigcup_{1 \leq i \leq n} H_i$;
- given a state $\langle s_1, \ldots, s_n \rangle$, an action a and a target distribution μ, there is a transition $\langle s_1, \ldots, s_n \rangle \xrightarrow{a} \mu$ if and only if μ is of the form $\mu_1 \times \ldots \times \mu_n$ and for all $1 \leq i \leq n$,
  - either $a \in Act(P_i)$ and $s_i \xrightarrow{a} \mu_i$,
  - or $a \notin Act(P_i)$ and $\mu_i = (s_i \mapsto 1)$.

Notice $\|$ is commutative and associative for PIOAs (up to isomorphism).

The notion of (probabilistic) schedulers for a PIOA P is introduced as a means to resolve all nondeterministic choices in P. Each scheduler consists of an input component and an output component. Given a finite history of the automaton, the output scheduler chooses probabilistically the next locally controlled transition, whereas the input scheduler responds to inputs from the environment and chooses probabilistically a transition carrying the correct input symbol.

Definition 1. *An* input scheduler σ *for* P *is a function*

$$\sigma : \mathsf{Exec}^{<\omega}(P) \times I \longrightarrow \mathsf{Disc}(\rightarrow)$$

such that for all $\langle p, a \rangle \in \mathsf{Exec}^{<\omega}(P) \times I$ *and transitions* $(s \xrightarrow{b} \mu) \in \mathsf{Supp}(\sigma(p, a))$, *we have* $s = \mathsf{last}(p)$ *and* $b = a$. *An* output scheduler ρ *for* P *is a function*

$$\rho : \mathsf{Exec}^{<\omega}(P) \longrightarrow \mathsf{SubDisc}(\rightarrow)$$

such that for all $p \in \mathsf{Exec}^{<\omega}(P)$ *and transition* $(s \xrightarrow{a} \mu) \in \mathsf{Supp}(\rho(p))$, *we have* $s = \mathsf{last}(p)$ *and* $a \in L$. *An* I/O *scheduler for* P *is then a pair* $\langle \sigma, \rho \rangle$ *where* σ *is an input scheduler for* P *and* ρ *is an output scheduler for* P.

Notice input schedulers must return a discrete probability distribution, reflecting the requirement that each input issued by the environment is received

with probability 1. (This is always possible because of the input enabling assumption.) In contrast, output schedulers may choose to halt with an arbitrary probability θ by returning a proper sub-distribution whose total probability mass is $1-\theta$. Finally, we write $\sigma(p,a)(\mu)$ as a shorthand for $\sigma(p,a)(\mathsf{last}(p) \xrightarrow{a} \mu)$ and $\rho(p)(a,\mu)$ for $\rho(p)(\mathsf{last}(p) \xrightarrow{a} \mu)$.

Consider a closed PIOA P. Obviously, any I/O scheduler for P has a trivial input component (i.e., the empty function). Every output scheduler ρ thus induces a purely probabilistic behavior, which is captured by the following notion of probabilistic executions. The *probabilistic execution* induced by ρ is the function $Q_\rho : \mathsf{Exec}^{<\omega}(P) \longrightarrow [0,1]$ defined recursively by:

- $Q_\rho(s^0) := 1$, where s^0 is the initial state of P;
- $Q_\rho(p') := Q_\rho(p) \cdot \rho(p)(a,\mu) \cdot \mu(s')$, where p' is of the form $pa\mu s'$.

A probabilistic execution Q_ρ induces a probability space over the sample space $\Omega_P := \mathsf{Exec}(P)$ as follows. Let $\sqsubseteq$ denote the prefix ordering on sequences. Each $p \in \mathsf{Exec}^{<\omega}(P)$ generates a *cone* of executions: $\mathbf{C}_p := \{p' \in \mathsf{Exec}(P) \mid p \sqsubseteq p'\}$. Let $\mathcal{F}_P$ denote the smallest σ-field generated by the collection $\{\mathbf{C}_p \mid p \in \mathsf{Exec}^{<\omega}(P)\}$. There exists a unique measure $\mathbf{m}_\rho$ on $\mathcal{F}_P$ with $\mathbf{m}_\rho[\mathbf{C}_p] = Q_\rho(p)$ for all p in $\mathsf{Exec}^{<\omega}(P)$; therefore Q_ρ gives rise to a probability space $(\Omega_P, \mathcal{F}_P, \mathbf{m}_\rho)$.

Trace distributions are obtained from probabilistic executions by removing non-visible elements. In our case, these are states, hidden actions and distributions of states. To state this precisely, we need the notion of minimal executions: a finite execution p of P is said to be *minimal* if every proper prefix of p has a strictly shorter trace. Notice, the empty execution (i.e., the sequence containing just the initial state) is minimal. Moreover, if p is nonempty and finite, then p is minimal if and only if the last transition in p has a visible action label. For each $\alpha \in Act(P)^{<\omega}$, let $\mathsf{tr}^{-1}_{\min}(\alpha)$ denote the set of minimal executions of P with trace α.

Now we define a lifting of the trace operator $\mathsf{tr} : \mathsf{Exec}^{<\omega}(P) \longrightarrow Act(P)^{<\omega}$. Given a function $Q : \mathsf{Exec}^{<\omega}(P) \longrightarrow [0,1]$, define $\mathsf{tr}(Q) : Act(P)^{<\omega} \longrightarrow [0,1]$ by

$$\mathsf{tr}(Q)(\alpha) := \sum_{p \in \mathsf{tr}^{-1}_{\min}(\alpha)} Q(p).$$

Given an output scheduler ρ of a closed PIOA P, the *trace distribution* induced by ρ (denoted D_ρ) is simply the result of applying tr to the probabilistic execution Q_ρ. That is, $D_\rho := \mathsf{tr}(Q_\rho)$. We often use variables D, D', etc. for trace distributions, thus leaving the scheduler ρ implicit.

Similar to the case of probabilistic executions, each D_ρ induces a probability measure on the sample space $\Omega := Act(P)^{\leq\omega}$. There the σ-field $\mathcal{F}$ is generated by the collection $\{\mathbf{C}_\alpha \mid \alpha \in Act(P)^{<\omega}\}$, where $\mathbf{C}_\alpha := \{\alpha' \in \Omega \mid \alpha \sqsubseteq \alpha'\}$. The measure $\mathbf{m}^\rho$ on $\mathcal{F}$ is uniquely determined by the equations $\mathbf{m}^\rho[\mathbf{C}_\alpha] = D_\rho(\alpha)$ for all $\alpha \in Act(P)^{<\omega}$.

In the literature, most authors define probabilistic executions (resp. trace distributions) to be the probability spaces $\langle \Omega_P, \mathcal{F}_P, \mathbf{m}_\rho \rangle$ (resp. $\langle \Omega, \mathcal{F}, \mathbf{m}^\rho \rangle$). Here we find it more natural to reason with the functions Q_ρ and D_ρ, rather

than the induced measures. We refer to [Seg95] for these alternative definitions and proofs that they are equivalent to our versions.

3 Switched Probabilistic I/O Automata

As we argued in Section 1, one must distinguish between global and local nondeterministic choices and must resolve them separately. This section describes our solution, namely, an explicit mechanism of control exchange among parallel components. The presentation is organized as follows: (i) first we define *pre-switched automata*, where we describe control action signatures and the Boolean-valued state variable active; (ii) then we introduce the notion of *input well-behaved executions* of a pre-switched automaton and state four axioms defining *switched automata*; (iii) finally, we introduce the notion of a *scheduled automaton*, essentially a switched automaton paired with an I/O scheduler.

For technical simplicity, we assume a universal set Act of action symbols such that $Act(P) \subseteq Act$ for every PIOA P. Moreover, Act is partitioned into two sets: $BAct$ (*basic actions*) and $CAct$ (*control actions*). Both sets are assumed to be countably infinite, so we can rename hidden actions using fresh symbols whenever necessary (cf. Section 5).

Definition 2. *A pre-switched automaton P is a PIOA endowed with a function* $\mathsf{active} : \mathsf{States}(P) \longrightarrow \{0,1\}$ *and a set* $Sync \subseteq O \cap CAct$ *of* synchronized control actions.

We use variables P, Q, etc. to denote pre-switched automata. Given a pre-switched automaton P, we further classify its action symbols:

- $BI := I \cap BAct$ (*basic inputs*);
- $BO := O \cap BAct$ (*basic outputs*);
- $CI := I \cap CAct$ (*control inputs*);
- $CO := (O \cap CAct) \setminus Sync$ (*control outputs*).

Essentially, we have a partition $\{BI, BO, H, CI, CO, Sync\}$ of $Act(P)$. We say that P is *initially active* if $\mathsf{active}(s^0) = 1$. Otherwise, it is *initially inactive*.

As described in Section 1, the Boolean-valued function active on the states of P indicates whether P is active or inactive, while control actions allow P to exchange control with its environment. The designation of synchronized control actions helps to achieve the "handshake" condition on control synchronizations: whenever we compose two automata, we classify the shared control actions as "synchronized", so that they are no longer available for further synchronization with a third component. This is made precise in the definitions of compatibility and composition for pre-switched automata.

A finite set of pre-switched automata $\{P_1, \ldots, P_n\}$ is said to be *compatible* if (i) $\{P_1, \ldots, P_n\}$ is a compatible set of PIOAs; (ii) for all $i \neq j$, $Act(P_i) \cap Sync_j = CI_i \cap CI_j = \emptyset$; (iii) at most one P_i is initially active. Notice that such a set is compatible if and only if for all $i \neq j$, P_i and P_j are compatible. The *parallel composition* of $\{P_1, \ldots, P_n\}$, denoted $\|_{1 \leq i \leq n} P_i$, is the result of composing $P_1, \ldots, P_n$ as PIOAs, together with:

– $Sync := \bigcup_{1 \le i \le n} Sync_i \cup \bigcup_{1 \le i,j \le n} (CI_i \cap CO_j)$;
– $\mathsf{active}(s_1, \ldots, s_n) = 1$ if and only if for some i, $\mathsf{active}_i(s_i) = 1$.

Clearly, the composite $\|_{1 \le i \le n} P_i$ is again a pre-switched automaton. In the binary case, we write $P_1 \| P_2$ as shorthand for $\|_{1 \le i \le 2} P_i$. Observe that $P_1 \| P_2 \cong P_2 \| P_1$; that is, composition of pre-switched automata is commutative up to isomorphism. Next we check that composition is also associative on the class of pre-switched automata.

Lemma 1. *Let P_1, P_2 and P_3 be pre-switched automata. Assume P_1 is compatible with P_2, and P_3 is compatible with $P_1 \| P_2$. Then P_2 is compatible with P_3, and P_1 is compatible with $P_2 \| P_3$. Moreover, $(P_1 \| P_2) \| P_3 \cong P_1 \| (P_2 \| P_3)$.*

Recall that switched automata are intended to be composed in such a way that at most one component is active at any point of an execution. In particular, any environment automaton must also follow the rules of control exchange; that is, after activating some system component, the environment must itself become inactive. This leads to the definition of *input well-behavedness*. Let P be a pre-switched automaton. An input transition $s \xrightarrow{a} \mu$ is *well-behaved* if $\mathsf{active}(s) = 0$. An execution p of P is *input well-behaved* if all input transitions occurring in p are well-behaved. Let $\mathsf{Exec}^{<\omega}_{\mathsf{iwb}}(P)$ denote the set of finite, input well-behaved executions of P. Moreover, we say that a state s is *input well-behaved reachable*, notation $\mathsf{iwbr}(s)$, if there exists an input well-behaved execution p such that $s = \mathsf{last}(p)$. Clearly, the empty execution is input well-behaved and thus the initial state is always input well-behaved reachable. If P is closed (i.e., $I = \emptyset$), then every execution of P is trivially input well-behaved and every reachable state is input well-behaved reachable. We are now prepared to define the notion of switched probabilistic I/O automata.

Definition 3. *A* switched (probabilistic I/O) automaton *is a pre-switched automaton P that satisfies the following axioms.*

$$s \xrightarrow{a} \mu \ \wedge \ \mathsf{active}(s) = 0 \quad \Rightarrow \quad a \in I \tag{1}$$

$$s \xrightarrow{a} s' \ \wedge \ a \in CI \quad \Rightarrow \quad \mathsf{active}(s') = 1 \tag{2}$$

$$s \xrightarrow{a} s' \ \wedge \ a \notin CI \cup CO \quad \Rightarrow \quad \mathsf{active}(s) = \mathsf{active}(s') \tag{3}$$

$$\mathsf{iwbr}(s) \ \wedge \ s \xrightarrow{a} s' \ \wedge \ a \in CO \quad \Rightarrow \quad \mathsf{active}(s') = 0 \tag{4}$$

These four axioms formalize our intuitions about control passage. Axiom (1) requires all inactive states to be quiescent. Axioms (2) and (4) say that control inputs lead to active states and control outputs to inactive states. Axiom (3) says that non-control transitions and synchronized control transitions do not change the activity status. Together, they describe an "activity cycle" for the automaton P: (i) while in inactive mode, P does not enable locally controlled transitions, although it may still receive inputs from its environment; (ii) when P receives a control input it moves into active mode, where it may perform hidden or output transitions, possibly followed by a control output; (iii) via this control output P returns to inactive mode.

Notice that Axiom (4) is required for input well-behaved reachable states only. Without this relaxation, the composition of two switched automata may not satisfy Axiom (4).

We proceed to confirm that the class of switched automata is closed under the parallel composition operator for pre-switched automata. A set $\{P_1, \ldots, P_n\}$ of switched automata is *compatible* if the set of underlying pre-switched automata is compatible. Define the composite, $\|_{1 \le i \le n} P_i$, to be the result of composing the switched automata as pre-switched automata. The first three axioms can be verified by unfolding the definition of active in a composition and applying appropriate axioms for the components. Axiom (4) follows from Lemma 2 below. The proof is by induction on the length of executions and relies heavily on invariant-style reasoning based on the definition of input well-behaved executions and the axioms of switched automata.

Lemma 2. *Let $\{P_1, \ldots, P_n\}$ be a compatible set of switched automata. For each finite, input well-behaved execution p of $\|_{1 \le i \le n} P_i$, we have:*

(i) for all i, $\pi_i(p)$ is also input well-behaved;
(ii) there is at most one i such that $\mathsf{active}_i(\pi_i(\mathsf{last}(p))) = 1$.

To summarize, $\|_{1 \le i \le n}$ is a well-defined n-ary operator for switched automata and, in the binary case, associativity of $\|$ follows from Lemma 1.

Next we turn to scheduling decisions. The notion of I/O schedulers for switched automata is inherited from that of its underlying PIOA.

Definition 4. *A* scheduled *automaton is a triple $\langle P, \sigma, \rho \rangle$ such that P is a switched automaton and $\langle \sigma, \rho \rangle$ is an I/O scheduler for P.*

We use letters S, T, etc. to denote scheduled automata. For each $1 \le i \le n$, let S_i denote a scheduled automaton $\langle P_i, \sigma_i, \rho_i \rangle$. The set $\{S_i \mid 1 \le i \le n\}$ is said to be *compatible* if $\{P_i \mid 1 \le i \le n\}$ is compatible as a set of switched automata. Given such a compatible set of scheduled automata, we obtain its composite by combining the I/O schedulers $\{\langle \sigma_i, \rho_i \rangle \mid 1 \le i \le n\}$ into an I/O scheduler $\langle \sigma, \rho \rangle$ for the switched automaton $\|_{1 \le i \le n} P_i$.

Definition 5. *Suppose $\{S_i \mid 1 \le i \le n\}$ is a compatible set of scheduled automata, where $S_i = \langle P_i, \sigma_i, \rho_i \rangle$ for each i. We construct from this set a* composite *scheduled automaton $\|_{1 \le i \le n} S_i := \langle P, \sigma, \rho \rangle$ as follows.*

- $P := \|_{1 \le i \le n} P_i$.
- *For every finite execution p of P with $\mathsf{last}(p) = s$ and for every $a \in I$,*
  - $\sigma(p, a)(t \xrightarrow{b} \mu) := 0$ *if $t \ne s$ or $b \ne a$;*
  - *otherwise, $\sigma(p, a)(s \xrightarrow{a} \mu_0 \times \ldots \times \mu_n) := \Pi_i c_i$, where c_i equals*
    - *$\sigma_i(\pi_i(p), a)(\mu_i)$, if $a \in I_i$;*
    - *1, otherwise.*
- *For every finite execution p of P with $\mathsf{last}(p) = s$,*
  - $\rho(p)(t \xrightarrow{a} \mu) := 0$ *if p is not input well-behaved, $t \ne s$, or $a \notin L$;*

- *otherwise,* $\rho(p)(s \xrightarrow{a} \mu_0 \times \ldots \times \mu_n) := \Pi_i c_i$, *where* c_i *equals*
  - $\rho_i(\pi_i(p))(a, \mu_i)$, *if* $a \in L_i$;
  - $\sigma_i(\pi_i(p), a)(\mu_i)$ *if* $a \in I_i$;
  - 1, *otherwise.*

It is routine to check that $\sigma(p, a)$ is a discrete distribution for all $p \in \mathsf{Exec}^{<\omega}(P)$ and $a \in I$. Lemma 2 guarantees that, at the end of every input well-behaved finite execution p, there is at most one i such that component i enables a locally controlled transition. This allows us to conclude that $\rho(p)$ is a discrete sub-distribution for all $p \in \mathsf{Exec}^{<\omega}(P)$.

As usual, we write $S_1 \| S_2$ for $\|_{1 \leq i \leq 2} S_i$, provided S_1 and S_2 are compatible. Associativity of $\|$ for scheduled automata follows from that for switched automata and a routine check on the I/O schedulers. Finally, the notions of probabilistic executions and trace distributions for closed scheduled automata are inherited from those of PIOAs. In particular, we write Q_S (respectively, D_S) for the probabilistic execution (respectively, trace distribution) induced by the output scheduler of a closed scheduled automaton S.

4 Projection and Pasting

In this section, we study projection and pasting of probabilistic behaviors. Such results are essential elements in constructing a compositional modeling framework. We begin by introducing the notion of regular executions, which will be used to define pseudo trace distributions for automata with open inputs. In Lemma 5, we prove that the pseudo distribution of a composite is uniquely determined by those of its components. Finally, we prove the main pasting lemma for closed automata (Lemma 7), which plays a crucial role in the proof of our main compositionality theorem (Theorem 1).

Given an execution p of a switched automaton P, we say that p is *regular* if it is both minimal and input well-behaved. Given a finite sequence α of visible actions in P, let $\mathsf{tr}^{-1}_{\mathsf{reg}}(\alpha)$ denote the set of regular executions of P with trace α. Notice that regularity coincides with minimality in case P is closed.

Lemma 3 states that, given a fixed trace, there is a bijective correspondence between the set of regular executions of the composite and the Cartesian product of the sets of regular executions of the two components.

Lemma 3. *Let* X *denote* $\mathsf{tr}^{-1}_{\mathsf{reg}}(\alpha)$ *in* $P_1 \| P_2$. *Let* Y *and* Z *denote* $\mathsf{tr}^{-1}_{\mathsf{reg}}(\pi_1(\alpha)$ *in* P_1 *and* $\mathsf{tr}^{-1}_{\mathsf{reg}}(\pi_2(\alpha))$ *in* P_2, *respectively. There exists an isomorphism* $\mathsf{zip} : Y \times Z \longrightarrow X$ *whose inverse is* $\langle \pi_1, \pi_2 \rangle$.

Next we introduce a notion of *pseudo* probabilistic execution for automata with open inputs. The definition itself is completely analogous to probabilistic executions for closed automata; however, a pseudo probabilistic execution does not always induce a probability measure, because it does not take into account the probabilities with which inputs are provided by the environment.

Definition 6. *Let $S = \langle P, \sigma, \rho \rangle$ be a scheduled automaton. Define the* pseudo probabilistic execution *Q of S as follows: for all finite executions p' of S,*

- *if p' is of the form s^0, where s^0 is the initial state of S, then $Q(p') := 1$;*
- *if p' is of the form $pa\mu s'$ with $a \in I$, then $Q(p') := Q(p) \cdot \sigma(p, a)(\mu) \cdot \mu(s')$;*
- *if p' is of the form $pa\mu s'$ with $a \in L$, then $Q(p') := Q(p) \cdot \rho(p)(a, \mu) \cdot \mu(s')$.*

Similarly, we define *pseudo* trace distributions.

Definition 7. *Let $S = \langle P, \sigma, \rho \rangle$ be a scheduled automaton. The* pseudo trace distribution *D of S is the function from $(Act(S) \setminus H_S)^{<\omega}$ to $[0, 1]$ given by $D(\alpha) := \sum_{p \in \mathsf{tr}^1_{\mathrm{reg}}(\alpha)} Q(p)$, where Q is the pseudo probabilistic execution of S.*

Notice that, if S is closed, then the pseudo probabilistic execution of S coincides with the probabilistic execution of S. Moreover, an execution of a closed automaton S is regular if and only if it is minimal, thus the pseudo trace distribution of S coincides with the trace distribution of S.

For the rest of this section, let S and T be a pair of compatible scheduled automata. Let $Q_{S\|T}$, Q_S and Q_T denote the pseudo probabilistic executions of $S\|T$, S and T, respectively. Similarly for pseudo trace distributions $D_{S\|T}$, D_S and D_T. Lemma 4 below says we can project a pseudo probabilistic execution of the composite to yield pseudo probabilistic executions of the components. The proof is routine, by induction on the length of executions. Lemma 5 then combines Lemma 3 and Lemma 4 to show the analogous projection result for pseudo trace distributions.

Lemma 4. *For all finite executions p of $S\|T$, we have $Q_{S\|T}(p) = Q_S(\pi_1(p)) \cdot Q_T(\pi_2(p))$.*

Lemma 5. *Let α be a finite sequence of visible action symbols of $S\|T$. Then $D_{S\|T}(\alpha) = D_S(\pi_1(\alpha)) \cdot D_T(\pi_2(\alpha))$.*

To prove the main pasting lemma, we need yet another technical result; namely, inputs must be received with probability 1. This can be viewed as "input enabling" in the probabilistic sense and it follows from basic properties of target distributions and input schedulers.

Lemma 6. *Let α be a finite sequence of visible action symbols of $S\|T$ and let $a \in Act(S\|T)$ be given. If a is not locally controlled by T, then $D_T(\pi_2(\alpha)) = D_T(\pi_2(\alpha a))$.*

Two switched/scheduled automata are said to be *comparable* if they have the same visible signature and their start states have the same status. We are now ready for the main pasting lemma.

Lemma 7 (Pasting). *Let S_1, S_2, T_1 and T_2 be scheduled automata satisfying (i) S_1 and S_2 are comparable; (ii) $\{S_1, T_1\}$, $\{S_2, T_2\}$ and $\{S_1, T_2\}$ are compatible sets; (iii) the pseudo trace distributions $D_{S_1\|T_1}$ and $D_{S_2\|T_2}$ coincide (denoted D). Then D also coincides with the pseudo trace distribution $D_{S_1\|T_2}$.*

5 Renaming and Hiding

In this section, we consider the standard renaming and hiding operators. We start with an equivalence relation on switched automata: $P_1 \equiv_R P_2$ just in case there exists a bijection $f : H_1 \longrightarrow H_2$ such that P_2 can be obtained from P_1 by replacing every hidden action symbol $a \in H_1$ by $f(a) \in H_2$ (notation: $P_2 = f(P_1)$).

It is routine to check this is in fact an equivalence relation. If $P_1 \equiv_R P_2$, we say that P_2 can be obtained from P_1 by *renaming of hidden actions*. This operation also induces an equivalence relation on scheduled automata: $\langle P_1, \sigma_1, \rho_1\rangle \equiv_R \langle P_2, \sigma_2, \rho_2\rangle$ just in case there exists a renaming function f such that $P_1 \equiv_R P_2$ via f and $\langle \sigma_2, \rho_2\rangle$ is obtained from $\langle \sigma_1, \rho_1\rangle$ via f and f^{-1} (notation: $S_2 = f(S_1)$).

The following lemma says, as long as the renaming operation does not introduce incompatibility of signatures, it does not affect the behavior of an automaton in a closing context.

Lemma 8. *Let S and C be compatible scheduled automata with $S\|C$ closed. Suppose $S \equiv_R S'$ via the renaming function $f : H \longrightarrow H'$ with H' disjoint from $Act(C)$. Then $\{S', C\}$ is closed and compatible and $D_{S\|C} = D_{S'\|C}$.*

Next we consider the issue of hiding output actions. Let Hide denote the standard hiding operator for PIOA. This is also an operator for switched automata, provided we hide only basic outputs and synchronized control actions.

Lemma 9. *Let P be a switched automaton and let $\Omega \subseteq BO \cup Sync$ be given. Then $\mathsf{Hide}_\Omega(P)$ is again a switched automaton.*

Notice that every I/O scheduler for P is an I/O scheduler for $\mathsf{Hide}_\Omega(P)$. Therefore Hide can be extended to scheduled automata:

$$\mathsf{Hide}_\Omega\langle P, \sigma, \rho\rangle := \langle \mathsf{Hide}_\Omega(P), \sigma, \rho\rangle.$$

In the rest of this section we investigate the effect of Hide_Ω on (pseudo) trace distributions. Let $S = \langle P, \sigma, \rho\rangle$ be a scheduled automaton with signature $\langle I, O, H\rangle$. For convenience, write P' for $\mathsf{Hide}_\Omega(P)$, O' for $O \setminus \Omega$, and tr' for the trace operator for $\mathsf{Hide}_\Omega(P)$. (If we view Hide_Ω as an operator on traces, then tr' is precisely $\mathsf{Hide}_\Omega \circ \mathsf{tr}$.)

Moreover, for all $\beta' \in (I \cup O')^{<\omega}$, let $\mathcal{M}_{\beta'}$ denote the set of all minimal (w.r.t. $\sqsubseteq$) traces in $\mathsf{Hide}_\Omega^{-1}(\beta')$. That is, if β' is empty, then $\mathcal{M}_{\beta'}$ is the singleton set containing the empty trace ϵ; otherwise,

$$\mathcal{M}_{\beta'} := \{\beta \in (I \cup O)^{<\omega} \mid \mathsf{Hide}_\Omega(\beta) = \beta' \text{ and the last symbol on } \beta \text{ is not in } \Omega.\}$$

We make a simple observation about minimal executions of P and those of P'.

Lemma 10. *For all $\beta' \in (I \cup O')^{<\omega}$, the following two sets are equal:*

- $X := \bigcup_{\beta \in \mathcal{M}_{\beta'}} \{p \in \mathsf{Exec}^{<\omega}(P) \mid \mathsf{tr}(p) = \beta$ *and p minimal w.r.t.* $\mathsf{tr}\}$;

– $Y := \{p \in \mathsf{Exec}^{<\omega}(P') \mid \mathsf{tr}'(p) = \beta' \text{ and } p \text{ minimal w.r.t. } \mathsf{tr}'\}$.

Now consider the pseudo trace distribution D_S. Define the effect of Hide_Ω on D_S to be the following function from $O'^{<\omega}$ to $[0,1]$:

$$\mathsf{Hide}_\Omega(D_S)(\beta') := \sum_{\beta \in \mathcal{M}_{\beta'}} D_S(\beta).$$

We have the following corollary of Lemma 10.

Corollary 1. *The pseudo trace distribution of* $\mathsf{Hide}_\Omega(S)$ *is precisely* $\mathsf{Hide}_\Omega(D_S)$. *That is,* $D_{\mathsf{Hide}_\Omega(S)} = \mathsf{Hide}_\Omega(D_S)$.

Finally, we consider the effect of hiding in a parallel composition. We claim that the act of hiding in one component does not affect the behavior of the other, as long as the actions being hidden in the first component are not observable by the second component. This idea is captured in the following lemma, which follows from Corollary 1 and Lemma 5.

Lemma 11. *Let* S_1, S_2, T *be scheduled automata satisfying: (i)* S_1 *and* S_2 *are comparable and (ii)* T *is compatible with* S_1 *and* S_2. *Let* $\Omega \subseteq O_T$ *be given and let* T' *denote* $\mathsf{Hide}_\Omega(T)$. *If* T' *is compatible with* S_1 *(and thus with* S_2*), then*

$$D_{S_1 \| T} = D_{S_2 \| T} \Leftrightarrow D_{S_1 \| T'} = D_{S_2 \| T'}.$$

6 Probabilistic Systems

In this section, we give a formal definition of our implementation preorder and prove compositionality. The basic approach is to model a system as a switched PIOA together with a set of I/O schedulers. Observable behavior is then defined in terms of trace distributions induced by the prescribed schedulers.

Formally, a *probabilistic system* $\mathcal{P}$ is a set of scheduled automata that share a common underlying switched automaton. (Equivalently, a probabilistic system is a pair $\langle P, \mathcal{S}\rangle$ where P is a switched automaton and $\mathcal{S}$ is a set of I/O schedulers for P.) Such a system is *full* if $\mathcal{S}$ is the set of all possible I/O schedulers for P.

Two probabilistic systems $\mathcal{P}_1 = \langle P_1, \mathcal{S}_1\rangle$ and $\mathcal{P}_2 = \langle P_2, \mathcal{S}_2\rangle$ are *compatible* just in case P_1 is compatible with P_2. The *parallel composite* of $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted $\mathcal{P}_1 \| \mathcal{P}_2$, is the probabilistic system: $\{S_1 \| S_2 \mid S_1 \in \mathcal{P}_1 \text{ and } S_2 \in \mathcal{P}_2\}$. Notice the underlying automaton of the composite is $P_1 \| P_2$.

Let S be a scheduled automaton. A *context* for S is a scheduled automaton C such that (i) C is compatible with S; (ii) S and C have complementary signatures (i.e., $I_C = O_S$ and $I_S = O_C$). Given probabilistic system $\mathcal{P} = \langle P, \mathcal{S}\rangle$, we say that D is a *trace distribution of* $\mathcal{P}$ just in case there exist scheduled automata $S \in \mathcal{P}$ and context C for S such that $D = D_{S\|C}$. We write $\mathsf{td}(\mathcal{P})$ for the set of trace distributions of $\mathcal{P}$.

Two probabilistic systems are *comparable* whenever the underlying switched automata are comparable. Given comparable systems $\mathcal{P}_1$ and $\mathcal{P}_2$, we define the

trace distribution preorder by: $\mathcal{P}_1 \leq_{\mathsf{td}} \mathcal{P}_2$ whenever $\mathsf{td}(\mathcal{P}_1) \subseteq \mathsf{td}(\mathcal{P}_2)$. We are now ready to present our main theorem, namely, that the trace distribution preorder for probabilistic systems is compositional.

Theorem 1. *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be comparable probabilistic systems with $\mathcal{P}_1 \leq_{\mathsf{td}} \mathcal{P}_2$. Suppose $\mathcal{P}_3$ is compatible with both $\mathcal{P}_1$ and $\mathcal{P}_2$. Then $\mathcal{P}_1 \| \mathcal{P}_3 \leq_{\mathsf{td}} \mathcal{P}_2 \| \mathcal{P}_3$.*

7 PIOA Revisited

Before concluding, we give an example in which switched automata are used to obtain a new trace-based semantics for general PIOAs. The idea is to convert a general PIOA to a switched PIOA by adding control actions and activity classification. We then hide all control actions in trace distributions generated by the resulting switched PIOA. In many cases, this yields a set of trace distributions strictly smaller than that considered by Segala [Seg95].

Let P be a PIOA and assume $Act(P) \subseteq BAct$. Let $\mathsf{go}, \mathsf{done} \in CAct$ be fresh symbols and let b_0 be a Boolean value. The *switch extension* of P with go, done and initialization b_0 (notation: $\mathcal{E}(P, \mathsf{go}, \mathsf{done}, b_0)$), is the switched automaton P' constructed as follows:

- $\mathsf{States}(P') = \mathsf{States}(P) \times \{0,1\}$ and the start state of P' is $\langle s^0, b_0 \rangle$;
- $I' = I \cup \{\mathsf{go}\}$, $O' = O \cup \{\mathsf{done}\}$, and $Sync' = \emptyset$;
- $\mathsf{active}'(\langle s, b \rangle) = b$ for $b \in \{0,1\}$;
- the transition relation is the union of the following:
  - $\{\langle \langle s, 1 \rangle, a, \mu^1 \rangle \mid s \xrightarrow{a} \mu \text{ in } P\}$,
  - $\{\langle \langle s, 0 \rangle, a, \mu^0 \rangle \mid s \xrightarrow{a} \mu \text{ in } P \text{ and } a \in I\}$,
  - $\{\langle \langle s, b \rangle, \mathsf{go}, (\langle s, 1 \rangle \mapsto 1) \rangle \mid s \in \mathsf{States}(P) \text{ and } b \in \{0,1\}\}$,
  - $\{\langle \langle s, 1 \rangle, \mathsf{done}, (\langle s, 0 \rangle \mapsto 1) \rangle \mid s \in \mathsf{States}(P)\}$,

  where μ^b denotes the distribution that assigns probability $\mu(t)$ to $\langle t, b \rangle$ and 0 to $\langle t, 1-b \rangle$.

Roughly speaking, P' is obtained from P by (i) adding a Boolean flag active' to each state; (ii) enabling locally controlled transitions only if $\mathsf{active}' = 1$; and (iii) adding go and done transitions which update active' appropriately. It is not hard to check that P' satisfies all axioms of switched automata. Moreover, the pair $\langle \mathsf{go}, \mathsf{done} \rangle$ can be easily generalized to a pair of disjoint sets of control actions.

Given any two compatible PIOAs, we can always extend them with complementary control actions and initialization statuses, resulting in a pair of compatible switched automata. As an example, we consider the automata Late and Toss in Figure 1. Actions a, b and c are considered outputs of Late, whereas action a is an input of Toss and actions e and f are outputs of Toss. The following diagrams illustrate $\mathcal{E}(\mathsf{Late}, \mathsf{go}, \mathsf{done}, 1)$ and $\mathcal{E}(\mathsf{Toss}, \mathsf{done}, \mathsf{go}, 0)$. For a clearer picture, we have omitted the probabilities on the input a-transition in Toss, as well as all nonessential input loops. The active region, which is identical to the

original PIOA, is drawn in the foreground. The inactive region, in which all locally controlled transitions are removed, is in the background. Each two-headed arrow indicates a control output from active to inactive and a control input from inactive to active.

Now consider the problematic trace distribution D_0 of $\mathsf{Late} \,\|\, \mathsf{Toss}$, as described in Section 1. Let $\mathcal{P}_1$ and $\mathcal{P}_2$ denote the full probabilistic systems on $\mathcal{E}(\mathsf{Late}, \mathsf{go}, \mathsf{done}, 1)$ and $\mathcal{E}(\mathsf{Toss}, \mathsf{done}, \mathsf{go}, 0)$, respectively. As we compose these two systems, D_0 is no longer a trace distribution of $\mathcal{P}_1 \| \mathcal{P}_2$ (even after hiding go and done), because I/O schedulers in $\mathcal{P}_1$ have no way of knowing whether action d or action e was performed by $\mathcal{P}_2$, thus they cannot establish the correlations between actions d and b, and between actions e and c.

This leads to our proposal of a new notion of visible behaviors for PIOA. Let P be a PIOA and let $\mathcal{P}$ be the full probabilistic system on $\mathcal{E}(P, \mathsf{go}, \mathsf{done}, 0)$. A PIOA E is a *context* for P if $I_E = O_P$, $O_E = I_P$, and E is compatible with P. For each such E, write $\mathcal{P}_E$ for the full probabilistic system on $\mathcal{E}(E, \mathsf{done}, \mathsf{go}, 1)$. We say that D is a *trace distribution* of P if there exists a context E for P such that $D \in \mathsf{td}(\mathsf{Hide}_{\{\mathsf{go},\mathsf{done}\}}(\mathcal{P} \| \mathcal{P}_E))$, where Hide is lifted from scheduled automata to probabilistic systems.

8 Conclusions and Further Work

Our ultimate goal, of course, is to obtain a compositional semantics for PIOAs. The notion of switch extensions opens up an array of new options for that end. A promising approach is to model each system as a finite set of PIOAs, rather than a single PIOA. Composition is taken to be set union, representing the act of placing two sets of processes in the same computing environment. Behavior is then defined in terms of switch extensions, which instantiate the system with a particular network topology for control passage. In that case, a behavior of a finite set $\mathcal{F}$ is determined by (i) a context E for $\mathcal{F}$; (ii) a combination of switch extensions of $\mathcal{F} \cup \{E\}$; (iii) a combination of I/O schedulers for these switch extensions. By ranging over all contexts and all extension-scheduler combinations, we capture all possible behaviors of the system represented by $\mathcal{F}$.

In other future work, we plan to apply our theory of composition for PIOAs to the task of verifying security protocols. For example, we will try to model typical Oblivious Transfer protocols within the PIOA framework and verify correctness in the style of Canetti's Universal Composability [Can01]. We will also explore the use of PIOAs as a semantic model for the probabilistic polynomial time process calculus of Lincoln, Mitchell, Mitchell and Scedrov [LMMS98].

References

[Agg94] S. Aggarwal. Time optimal self-stabilizing spanning tree algorithms. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, May 1994. Available as Technical Report MIT/LCS/TR-632.

[Can01] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computing*, pages 136–145, 2001.

[dAHJ01] L. de Alfaro, T.A. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. In K.G. Larsen and M. Nielsen, editors, *Proceedings CONCUR 01,* Aalborg, Denmark, August 20-25, 2001, volume 2154 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2001.

[LMMS98] P. Lincoln, J.C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic polytime framework for protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 112–121, 1998.

[LSS94] N.A. Lynch, I. Saias, and R. Segala. Proving time bounds for randomized distributed algorithms. In *Proceedings of the 13th Annual ACM Symposium on the Principles of Distributed Computing*, pages 314–323, Los Angeles, CA, August 1994.

[LSV03] N.A. Lynch, R. Segala, and F.W. Vaandrager. Compositionality for probabilistic automata. In R. Amadio and D. Lugiez, editors, *Proceedings 14th International Conference on Concurrency Theory (CONCUR 2003),* Marseille, France, volume 2761 of *Lecture Notes in Computer Science*, pages 208–221. Springer-Verlag, September 2003.

[LT89] N.A. Lynch and M.R. Tuttle. An introduction to input/output automata. *CWI Quarterly*, 2(3):219–246, September 1989.

[PSL00] A. Pogosyants, R. Segala, and N.A. Lynch. Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. *Distributed Computing*, 13(3):155–186, 2000.

[Seg95] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1995. Available as Technical Report MIT/LCS/TR-676.

[Sto02] M.I.A. Stoelinga. An introduction to probabilistic automata. *Bulletin of the European Association for Theoretical Computer Science*, 78:176–198, October 2002.

[SV99] M.I.A. Stoelinga and F.W. Vaandrager. Root contention in IEEE 1394. In J.-P. Katoen, editor, *Proceedings 5th International AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems,* Bamberg, Germany, volume 1601 of *Lecture Notes in Computer Science*, pages 53–74. Springer-Verlag, 1999.

[SV04] A. Sokolova and E.P. de Vink. Probabilistic automata: system types, parallel composition and comparison. In C. Baier et al., editor, *Validation of Stochastic Systems*, volume 2925 of *Lecture Notes in Computer Science*, pages 1–43. Springer-Verlag, 2004.

[WSS94] S.-H. Wu, S. A. Smolka, and E. W. Stark. Composition and behaviors of probabilistic i/o automata. In B. Jonsson and J. Parrow, editors, *Proceedings CONCUR 94,* Uppsala, Sweden, volume 836 of *Lecture Notes in Computer Science*, pages 513–528. Springer-Verlag, 1994.